A Soft-Input Soft-Output APP Module for Iterative Decoding of Concatenated Codes

S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara

Abstract — Concatenated coding schemes consist of the combination of two or more simple *constituent encoders* and interleavers. The parallel concatenation known as "turbo code" has been shown to yield remarkable coding gains close to theoretical limits, yet admitting a relatively simple iterative decoding technique. The recently proposed serial concatenation of interleaved codes may offer superior performance to that of turbo codes. in both coding schemes, the core of the iterative decoding structure is a soft-input soft-output (S1S0) *a posteriori* probability (APP) module. In this letter, wc describe the S1S0 APP module that updates the *a posteriori* probabilities corresponding to the input and the output bits, of a code, and show how to embed it into an iterative decoder for a new hybrid concatenation of three codes, to fully exploit the benefits of the proposed S1S0 APP module.

Introduction — concatenated coding schemes have been studied by Forney [1] as a class of codes whose probability of error decreased exponentially at rates less than capacity, while decoding complexity increased only algebraically. Initially motivated only by theoretical research interests, concatenated codes have since then evolved as a standard for those applications where very high coding gains arc needed, such as (deep-)spacc applications.

The recent proposal of "turbo codes" [2], with their astonishing performance close to the theoretical Shannon capacity limits, have once again shown the great potential of coding schemes formed by two or more codes working in a concurrent way. Turbo codes are *parallel concatenated convolutional codes*, where the information bits arc encoded twice. Once by a recursive systematic convolutional code acting cm the original information sequence, and a second time by a (possibly different) recursive convolutional code acting on a permuted information sequence. The code sequences arc formed by the information bits, followed by the parity check bits generated by both encoders. Using the same ingredients, namely convolutional encoders and interleavers, *serially concatenated convolutional codes* have been shown to yield performance comparable, and in some cases superior, to turbo codes [3]. A third choice is a hybrid concatenation of three or more codes. In this letter, wc consider as an example of hybrid concatenation, the parallel concatenation of a convolutional code with two serially concatenated convolutional codes.

All concatenated coding schemes admit a suboptimum decoding scheme based on the iterative use of *a posteriori* probability algorithms [4] applied to each constituent code. The purpose of this letter is the description of a soft-input soft-output module (denoted by S1S0) that implements the APP algorithm in its basic form for the iterative decoding of a concatenated coding scheme. (Wc prefer to usc the APP terminology instead of MAP (maximum *a posteriori*) *since* the S1S0 module is just computing probabilities rather than their maximum.)

The encoder — The S1S0 module is a four-port device, with two inputs and two outputs. It accepts as inputs the probability distributions of the information and code symbols labeling the edges of the code trellis, and forms as outputs an update of these distributions based upon the code constraints. The algorithm for the S1S0 module works on the trellis representation of the code (every code admits a trellis representation). It can be a time-invariant or time-

varying trellis, and thus the algorithm can be used for both block and convolutional codes. In the following, for simplicity of exposition, wc will refer to the case of *binary time-invariant convolutional codes* with cock rate k_o/n_o.

Figure 1: *The trellis encoder*

In Fig. 1 wc show a *trellis encoder*, characterized by the following quantities. (Capital letters U, C, S, E will denote random variables, and lower case letters u, c,s, e their realizations. The subscript k will denote a discrete time, defined on the time index set K. The letters I, and O will refer to the input and output of the S1S0 module, respectivel y<)

1. U = $(U_k)_{k\in K}$ is the sequences of input symbols, defined over a time index set K (finite or infinite) and drawn from the alphabet $U = \{u_1, \ldots . u_{N_I}\}$. Each input symbol U_k consists of k_o bits U_k^j, j=1,2,..,k_o with realization $u^j \in$ {O, l}. To the sequence of input symbols, wc associate the sequence of *a priori* probability y distributions $\mathbf{P}(\mathbf{u}; I) = (P_k(u; I))_{k\in K}$, where $P_k(u; I) = \prod_{j=1}^{k_o} P_k(u^j; I)$.

2. $\mathbf{C} = (C_k)_{k\in K}$ is the sequences of output, or code, symbols, defined over the same time index set K, and drawn from the alphabet $\mathcal{C} = \{c_1, \ldots . c_{N_O}\}$. Each input symbol C_k consists of n_o bits C_k^j, j=1,2,..,n_o with realization $c^j \in \{0,1\}$. To the sequence of output symbols, we associate the sequence of *a priori* probability distributions $\mathbf{P}(\mathbf{c}; I) = (P_k(\mathrm{c}; I))_{k\in K}$, where $P_k(c; I)$ may be represented as $P_k(\mathbf{c}; I) = \prod_{j=1}^{n_o} P_k(c^j; I)$.

The assumption that *a priori* input distributions of symbols can be represented as a the product of marginal distributions of bits is valid when bit interleavers rather than symbol interleavers are used in an iterative decoding scheme for concatenated codes. One should not usc P_k(c; I) as a product for those encoders in a concatenated system where the output C in Fig.1 is connected to a nonbinary-input channel. For binary-input memoryless channels, the probability P_k(c; I) may be written as a product.

The **trellis section** -- The dynamics of a time-invariant convolutional code is completely specified by a single *trellis section*, which describes the transitions ("edges") between the states of the trellis at time instants k and k + 1.

A trellis section is characterized by: A set of N states $\mathcal{S} = \{s_1, \ldots . s_N\}$. The state of the trellis at time k is S_k = s, with s $\in \mathcal{S}$; A set of N . N_I edges obtained by the Cartesian product $\mathcal{E} = \mathcal{S} \times \mathcal{U} = \{e_1, \ldots, e_{N\cdot N_I}\}$ which represent all possible transitions between the trellis states.

To each edge $e \in \mathcal{E}$ the following functions are associated (see Fig. 2): The starting state $s^S(e)$ (the projection of e onto $\mathcal{S}$); The ending state s^E (c); The input symbol $u(e)$ (the projection of e onto U); The output symbol $c(e)$.

The relationship between these functions depends on the particular encoder. As an example, in the case of systematic encoders (s^E(e), $c(e)$) also identifies the edge since $u(e)$ is unique] y determined by $c(e)$. In the following, wc only assume that the pair $(s^S(e), u(e))$ uniquely identifies the ending state $s^E(e)$; this assumption is always fulfilled, as it is equivalent to say

that, given the initial trellis state, there is a one-to-one correspondence between input sequences and state sequences.

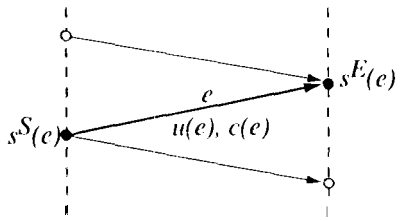

Figure 2: *An* ***edge*** *of the trellis section*

The S1S0 algorithm — The Soft-Input soft-Output (S1S0) module is a four port device that accepts at the input the sequences of probability distributions $\mathbf{P}(\mathbf{c}; I)$ and I'(u; I), and outputs the sequences of probability distributions **1'(c; O) and P(u;** O) based on its inputs and on its knowledge of the trellis section (or code, in general).

Figure 3: *The Soft-Input Soft-Output (SISO) module*

let the time index set $\boldsymbol{K = \{1, \ldots . n\}}$. The algorithm by which the S1S0 operates in evaluating the output distributions will be explained in two steps. First, we consider the following algorithm:

1. The output probability distributions $\tilde{P}_k(c^j; \mathrm{O})$ and $\tilde{P}_k(u^j; O)$ for the jth bit within each symbol at time k are computed as

$$\tilde{H}_{c^j} \sum_{e:C_k^j(e)=c^j} A_{k-1}[s^S(e)]P_k[u(e); I]P_k[c(e); I]B_k[s^E(e)] \tag{1}$$

and

$$\tilde{H}_u^j \sum_{e:U_k^j(e)=u^j} A_{k-1}[s^S(e)]P_k[u(e); I]P_k[c(e); I]B_k[s^E(e)] \tag{2}$$

respectively.

2. The quantities $A_k(\cdot)$ for $k = 1, \ldots, n$ and $B_k(\cdot)$ for $k = n-1, \ldots . 0$ arc obtained through the *forward* and *backward* recursions, respect i vel y, as

$$AL(s) = \sum_{e:s^k(e)=s} A_{k-1}[s^S(e)]P_k[u(e); I]P_k[c(e); I] \tag{3}$$

$$B_k(s) = \sum_{e:s^S(e)=s} B_{k+1}[s^E(e)]P_{k+1}[u(e); I]P_{k+1}[c(e); I] \tag{4}$$

with initial values $A_0(s) = 1$ if $s = S_0$ and $A_0(s) = O$ otherwise; B_n (s) = 1 if $s = S_n$ and B_n (.s) = O otherwise. The quantities $\tilde{H}_{c^j}, \tilde{H}_{u^j}$ arc normalization constants such that $\sum_{c^j} \tilde{P}_k(c^j; \boldsymbol{O}) = \boldsymbol{1}$ and $\sum_{u^j} \tilde{P}_k(u^j; \mathrm{O}) = 1$, respectively.

From expressions (1) and (2), it is apparent that the quantities $P_k[c^j(e);I]$ in the first equation and $P_k[u^j(e.);I]$ in the second do not depend on e, by definit ion of the summation indices, and thus can be extracted from the summations. Thus, defining the new quantities $P_k(c^j;O) \stackrel{\Delta}{=} H_{c^j}\frac{\tilde{P}_k(c^j;O)}{P_k(c^j;I)}$ and $P_k(u^j;O) \stackrel{\Delta}{=} H_{u^j}\frac{\tilde{P}_k(u^j;O)}{P_k(u^j;I)}$, where H_{c^j}, H_{u^j} arc normal izat ion constants such that $\sum_{c^j} P_k(c^j;O) = 1$ and $\sum_{u^j} P_k(u^j;O) = 1$, it can be easily verified that $P_k(c^j;O)$ and $P_k(u^j;0)$ can be obtained through the expressions

$$H_{c^j}\tilde{H}_{c^j}\sum_{e:C_k^j(e)=c^j} A_{k-1}[s^S(e)]P_k[u(e);I]\left(\prod_{\substack{i=1\\ i\neq j}}^{n_o} P_k[c^i(e);I]\right)B_k[s^E(e)] \qquad (5)$$

$$H_{u^j}\tilde{H}_{u^j}\sum_{e:U_k^j(e)=u^j} A_{k-1}[s^S(e)]\left(\prod_{\substack{i=1\\ i\neq j}}^{k_o} P_k[u^i(e);I]\right)P_k[c(e);I]B_k[s^E(e)] \qquad (6)$$

respectively, where the $A's$ and $B's$ satisfy the same recursions previously introduced in (3) and (4). For those encoders, in a concatenated coded scheme, whose outputs arc connected to the channel (as opposed to the input of another encoder), cq. (5) needs not be calculated. To keep the expressions general, as it is seen from (3), (4) and (6), P_k [c(e); I] is not represented as a product.

The new probability y distributions $P_k(c^j;O), P_k(u^j;O)$ arc computed based on the code constraints and obtained using the probability distributions of all bits of the sequence, except the distributions $P_k(c^j;I), P_k(u^j;I)$ of the j-th bit within the k-th symbol respectively. In the literature of "turbo decoding", $P_k(u^j;O), P_k$ (c-J: O) would be called *extrinsic bit information*. They represent the "added value" of the S1S0 module to the "*a priori*" distributions $P_k(u^j;I), P_k(c^j;I)$. Basing the S1S0 algorithm on $P_k(\cdot;O)$ instead of on $\tilde{P}_k(\cdot;O)$ simplifies the block diagrams, and related software and hardware, of the iterative decoding schemes. In addition, in the iterative decoding, the probability $P_k(\cdot;O)$, and not $\tilde{P}_k(\cdot;0)$, from one S1S0 module should be used as an input say $P_{k'}$(o; I) to another S1S0 module. For these reasons, we will consider as S1S0 algorithm the one expressed by (5) and (6). The S1S0 module is then represented as in Fig. 3. Applying the "log" operation to both sides of (3), (4), (5), and (6), results in the "Additive (log) APP S1S0".

Previously proposed algorithms were not in a form suitable to work with a general trellis code. Most of them assumed binary input symbols, some assumed also systematic codes, and none (not even the original BCJR algorithm [4]) could cope with a trellis having parallel edges. As it can be noticed from all summations involved in the equations that define the S1S0 algorithm, wc work on trellis edges, rather than on pairs of states, and this makes the algorithm completely general, and capable of coping with parallel edges and, also, encoders with rates greater than one, like those encountered in some concatenated schemes.

An application of the S1S0 module — We show in this section an example of application of the S1S0 module embedded into a proposed iterative decoding scheme, shown in Fig. 4, for decoding of a new hybrid concatenation of three convolutional codes, which is also shown in Fig. 4. The overall rate is 1/4, since the information bits of the systematic recursive parallel

encoder are not transmitted. As it is seen from Fig. 4, all four ports of the S1S0 outer module arc used.

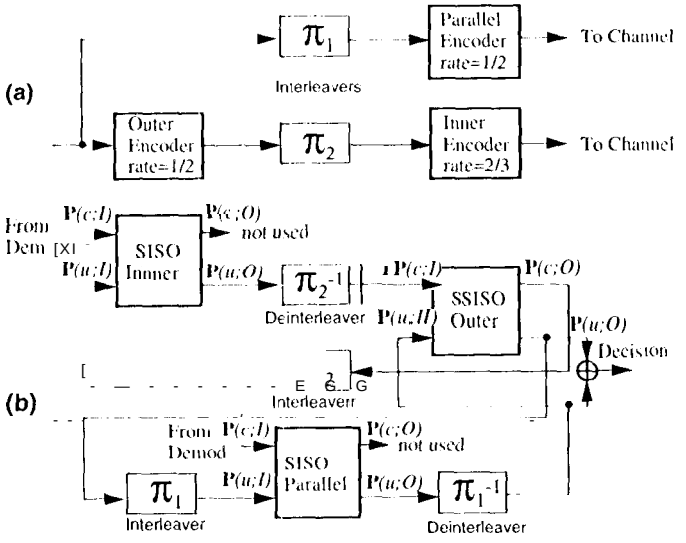

Figure 4: *(a) – A rate **1/4** Hybrid concatenated code, and (b) – **its iterative** decoding*

Acknowledgment — This research has been partially supported by NATO under Research Grant CRG 951208, and partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administrate ion.

S. Benedetto, and G. Montorsi *(Politecnico di Torino, Italy)*
D. Divsalar and F. Pollara *(Jet Propulsion Laboratory, California Institute of Technology, **USA**)*

References

[1] G. Il. Forney Jr., *Concatenated Codes*, M. I."l'., Cambridge, MA, 1966.

[2 C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: TLtrbo-Codes", in *Proceedings of ICC'93, Geneva, S* wit zerland, May 1993, pp. 1064-1070.

[3 S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding," *TDA Progress Report* 42-/26, *April–June /996,* Jet Propulsion Laboratory, Pasadena, California, pp. 1–26, August 15, 1996. http://edms-www.jpl.nasa.gov/tda/progress_report/42-1 26/1 26 D.pdf

[4] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate". *IEEE Transactions on In formation Theory*, pages *284-287,* March 1974.